ECE385

Fall 2021

Experiment 3

A Logic Processor

Ge Yuhao, Lou Haina
D231, Oct.10, 2021
TA: Chenhao Wang

1 Introduction

1.1 Summary

In this experiment, we design and build a bit-serial logic operation processor. We can load values into two 4-bit shift registers, and do some operations (NAND, NOR, XNOR, SET0, etc.) on them. We can also determine which register will the output be stored to through a routing unit.

1.2 Answers to pre-lab questions

Describe the simplest circuit that can optionally invert a signal.

We can simply used to "AND" gates and one "OR" gate to build the circuit. The SOP logic is:

$$Output = Input \cdot \overline{Invert} + \overline{Input} \cdot Invert$$

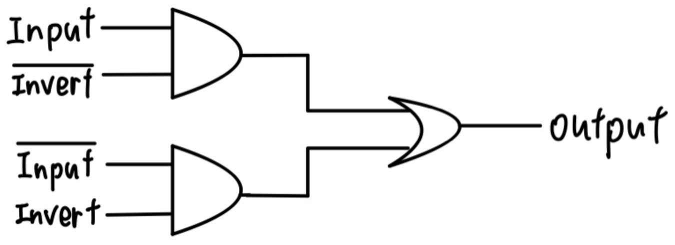

Figure 1: Signal Inverting Circuit

Explain how a modular design such as that presented above improves testability and cuts down development time.

In a modular design, we only care about the inputs and outputs of the module. For the above design, we can package it into a simple module with two inputs and one output and denote the module with name "signal inverting unit". Then each time we need to use such functionality, we can directly use the same module block into the circuit without caring about the inner structure of this module. If the module functions well as a single unit, it will probably error-free in the inserted circuit. Also, each time we want to modify the functionality of a module, all the parts of the circuit using this module will automatically change behavior accordingly, which can save a lot of time.

2 Design, document and build the circuit

Generally, we use NAND gate, D-Flip Flop, 4-bit shift register, 4-1 MUX, 4-bit counter to build the circuit, and we break out our circuit into four units: Register Unit, Computation Unit, Routing Unit, and Computation Unit.

2.1 Register Unit

The shift register unit consists of two 4-bit shift registers. One is $A_1A_2A_3A_4$ and the other is $B_1B_2B_3B_4$. The least significant bit of the regisyer will be connected to the computation unit as the computation unit can just process one bit at a time.

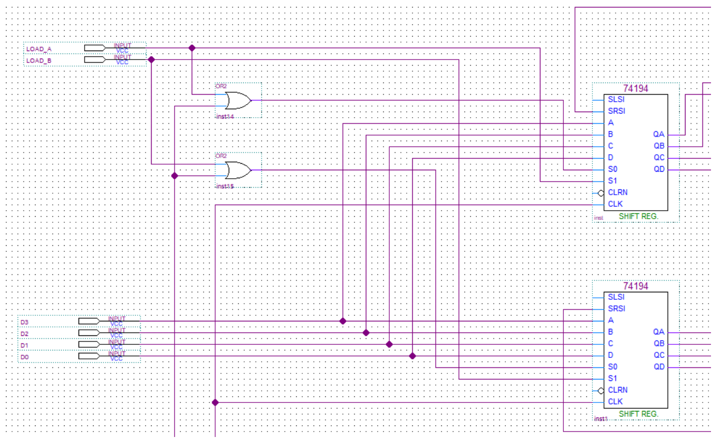

Figure 2: Register Unit

2.2 Computation Unit

The computation unit takes the least significant bit from both registers A0, B0 and performs 8 different operations using logic gates according to the input F_1, F_2, F_3.

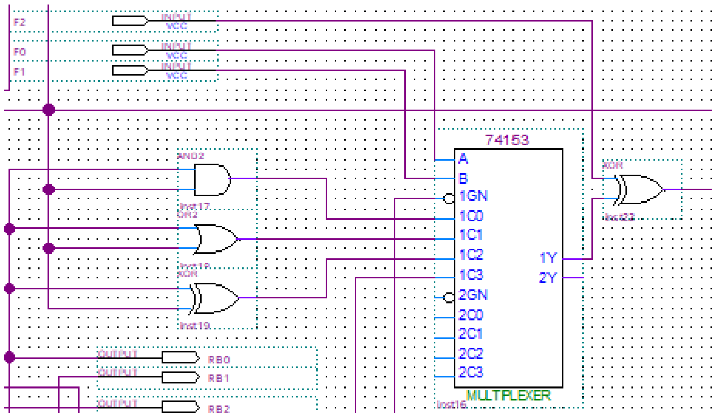

Figure 3: Register Unit

2.3 Routing Unit

The routing unit is made up of two 4-to-1 multiplexers (one for A', another for B').The select bit R_1R_0 will choose the value of A' and B' to go into the shift registers.

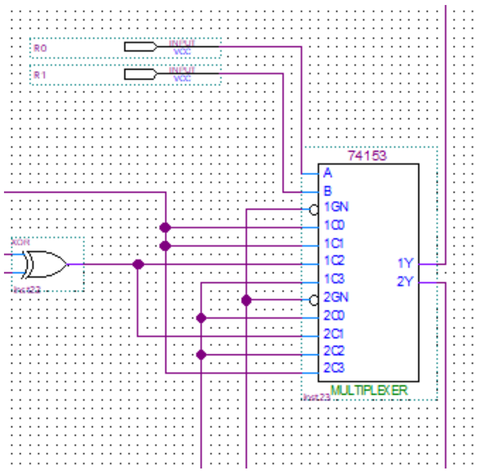

Figure 4: Register Unit

2.4 Control Unit

The control unit accept input like *LOAD A*, *LOAD B*, *Execute*, and will give output like *Shift*. A Mealy machine is used in this design and a 2-bits counter is used to simplify the circuit of control unit. The Detailed Circuit Schematic is at 5.2.

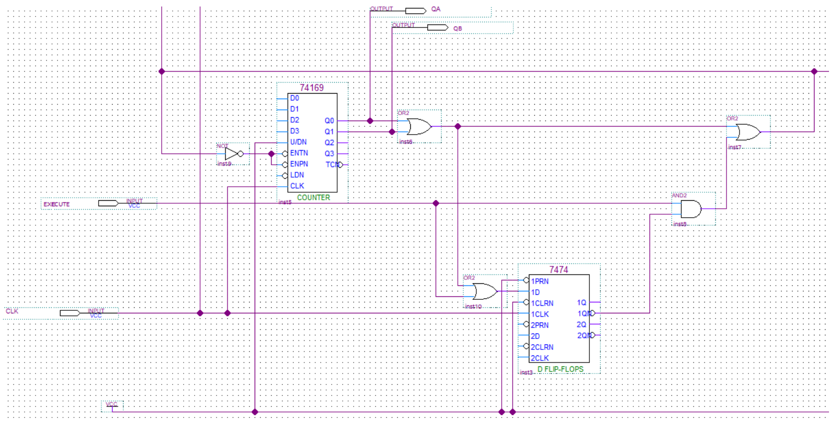

Figure 5: Control Unit

3 Operation of the logic processor

3.1 Describe the sequence of switches the user must flip to load data into the A and B registers

- The user must flip the switch to make *LOAD A* or *LOAD B* high, and flip the switches to set D_0, D_1, D_2, and $D3$ to the value that the user want to store into the register.

3.2 Describe the sequence of switches the user must flip to initiate a computation and routing operation

- The user must flip the switch to make *Excute* high to initiate a computation. The input "F" is used to denote which operation the user want to conduct. Below is a table which indicate the correspondence between the ”F” switches and the operation type. The design of the routing operation is similar.

F2	F1	F0	Output
0	0	0	A AND B
0	0	1	A OR B
0	1	0	A XOR B
0	1	1	1111
1	0	0	A NAND B
1	0	1	A NOR B
1	1	0	A XNOR B
1	1	1	0000

Table 1: Function Selection

R1	R0	A*	B*
0	0	A	B
0	1	A	F
1	0	F	B
1	1	B	A

Table 2: Routing Selection

Figure 6: Functions

4 Written description, block diagram and state machine diagram of logic processor

4.1 Written description: describe in words each block in the high-level diagram (a short paragraph for at least the register unit, computation unit, routing unit, and control unit)

- The register unit is made up of two 4-bit shift registers, which is used to store data that allows read and write. The control of these registers will be provided from the Control Unit.
- The computation unit is used to make the circuit capable of calculating eight different functions such as NAND, NOR, XNOR, SET0, etc.. The computation unit will accept input from the contents of RegA and RegB, and the function selection inputs F2, F1, F0. The unit will output the logical function f(A, B) specified by ¡F2, F1, F0¿ and will also output the A and B inputs unchanged. The three outputs will be fed to the Routing Unit.
- The routing unit is used to route the results of those operations in four different ways. The routing unit will accept the A, B, and f(A, B) as inputs, and will determine which signals to feed to the new A and new outputs based on the routing selection inputs R1, R0.
- The control unit will accept the following inputs: Load A, Load B, Execute, and the clock signal. The Load A and Load B inputs will perform parallel loads from the INPUT port (D3-D0) into the A and B registers. Execute tells the control unit that the select switches and the register contents are ready for execution and that the control unit should begin the computation cycle. The control unit then shifts the register unit the required number of times and halts until the next execution is requested. Obviously, some type of mechanism to keep track of the shifts will be required. A finite state machine will be implemented to serve as the control unit of the circuit.

4.2 A high-level block diagram.

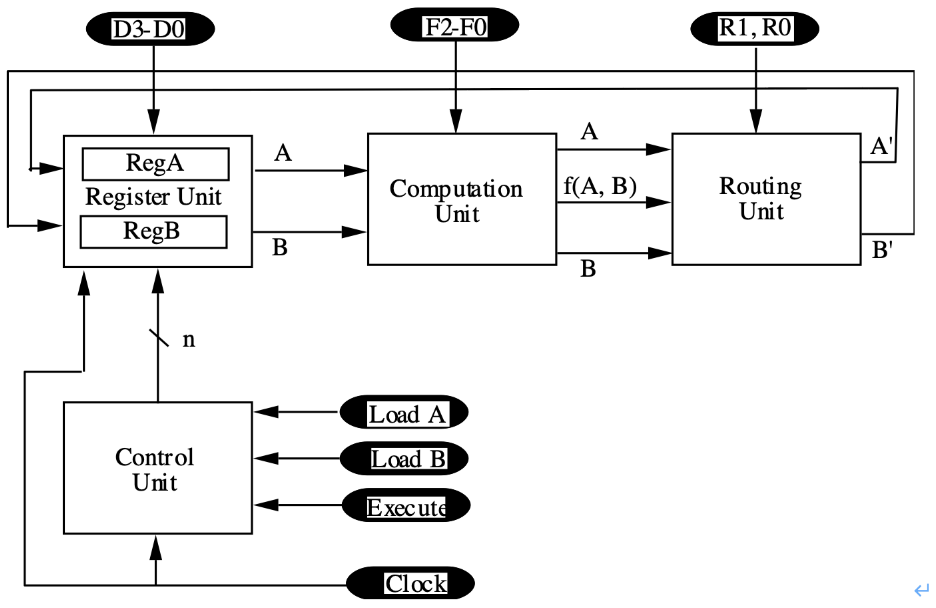

Figure 7: High-level Block Diagram

4.3 State Machine Diagram

- We use a Mealy machine in this lab which has fewer states.
- State "0" represents the state that the machine is at rest (not shifting), waiting to be executed, and the value in the binary flip-flop is 0 at this state. State "1" represents the state that the machine is working, namely shifting or halting, and the value in the binary flip-flop is 1 at this state.
- Arcs are labeled with input "Count"/"Execute" and output "Shift".

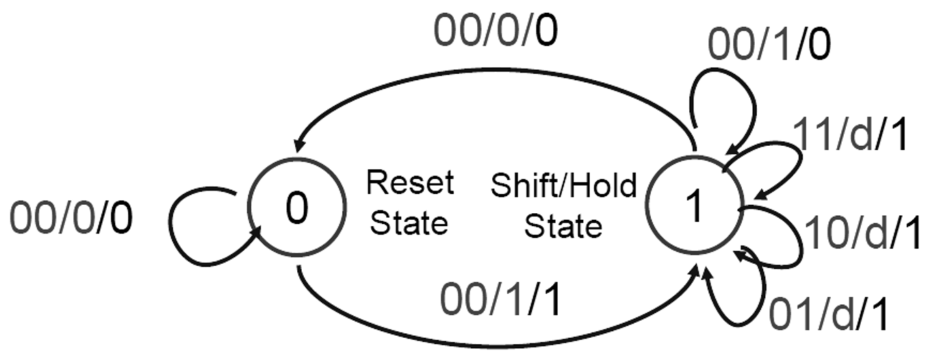

Figure 8: State Diagram

5 Design steps taken and detailed circuit schematic diagram

5.1 Written procedure of the design steps taken

k-maps for building control unit.

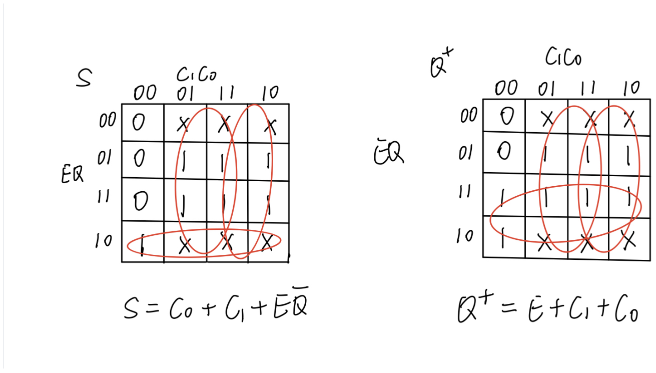

Figure 9: k-map for FSM

When we design control unit, we need to build Mealy or More machine, because Mealy machine has less states and only needs two bits to represent its state, we think it will be more convenient to write k-map and need less flip-flops, therefore we choose Mealy machine instead of More machine, and also it acts faster to inputs because it is triggered also by inputs. Besides, we use clock to make C1,C0 change one after one automatically instead of complex logic gates. Apart from this, to realize 8 functions in computation unit, we can choose 8-1 MUX or two 4-1 MUXs or use some logic to simplify hard design. We use XOR connect F2 input and the output by four functions (AND, OR, XOR, 1111).

5.2 Detailed Circuit Schematic

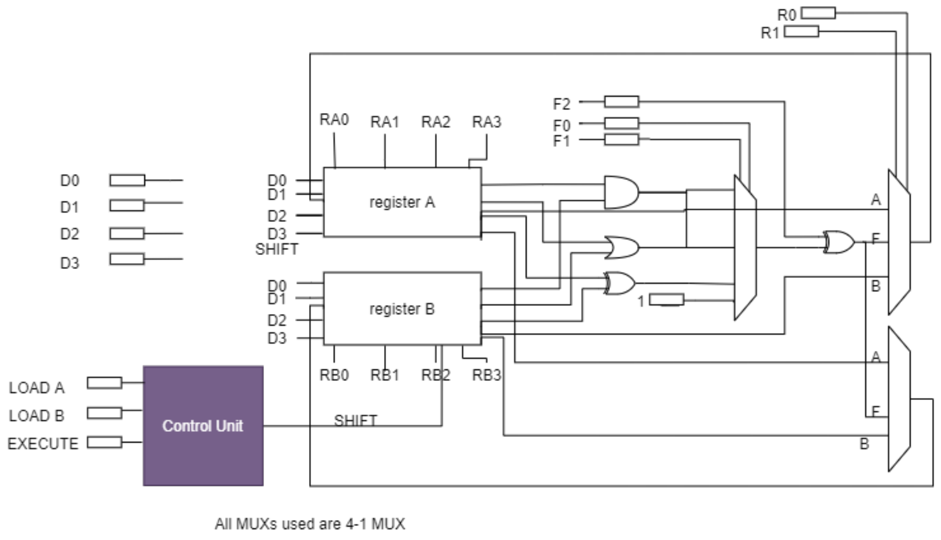

Figure 10: gate-level schematic

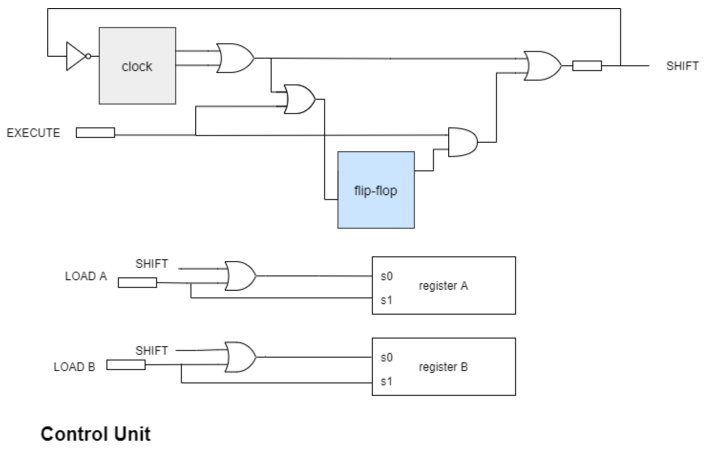

Figure 11: detailed schematic for control unit

6 Description of all bugs encountered, and corrective measures taken

Our design process is relatively smooth. We met some bugs when we connect clock, we connect wrong pins of 74169 and the output signal shift is wrong. Therefore, we set two outputs from clock directly and connect the EN to vcc, after checking the clock can work well, we connect it back into our circuit.

7 Conclusion

7.1 Summarize the lab in a few sentences

In this lab we design and build a bit-serial logic operation processor. We can load values into two 4-bit shift registers, and do some operations (NAND, NOR, XNOR, SET0, etc.) on them. We can also determine which register will the output be stored to through a routing unit. We use a Mealy machine to design the control unit, and a counter is used to reduce complexity of the control unit circuit.

7.2 Answer to all post-lab questions

Document changes to your design, explaining any difficulties you had in debugging your circuit.

When we design the control unit, we make bugs in connecting clock, we connect wrong pins of 74169 and the output signal shift is wrong. After disconnecting it and check its function, we know the correct pin connection.

Outline how the modular approach proposed in the pre-lab help you isolate design and wiring fault

We diveided our circuit design into four individual modules (Control unit, register unit, computation unit, and routing unit). We built them and test them individually to make sure that each of them is working well. Especially for the control unit which is the most complex, we first built it and tested if the output signal $Shift$ of this unit is correct, and we found that the $shift$ signal didn't last for 4 clock circle which means that the circuit is not correct. Then we debugged and corrected the circuit. After that, we connected the other three parts, and the whole circuit just worked within our expectations. If we didn't separate the module, the debugging process would be much more suffering.

Discuss the design process of your state machine, what are the tradeoffs of a Mealy machine vs a Moore machine?

We chose to use Mealy machine. Mealy machine generates output according to the input and the current state while Moore machine's output depends only on the current state. It seems that a Moore machine's output is much more straightforward, but this advantage is not that important. Instead, the Mealy machine is more suitable for the design. Firstly, Mealy machine has less states, and thus needs less wires and gates which may reduce the complexity of designing and debugging. Also, because we only need to use one bit to represent its state, only one flip-flop is needed, which save hardware components. Besides, it acts faster to inputs because it is triggered also by inputs.

8 Waveform generated by the standard testing input

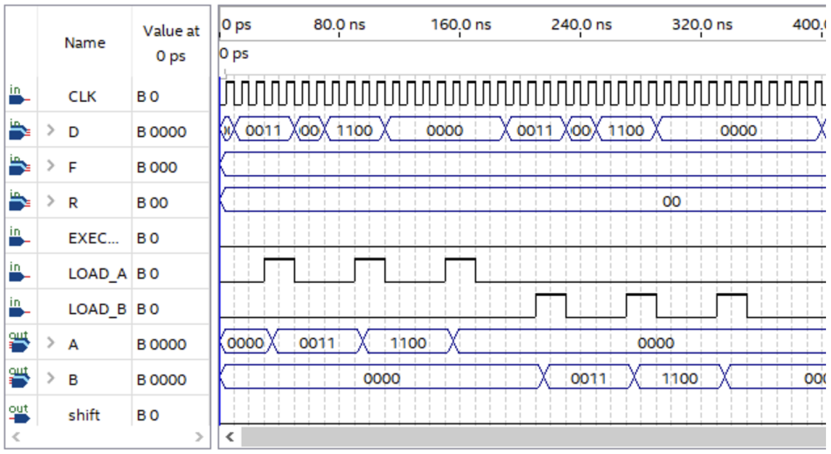

Figure 12: wave of 0-400ns

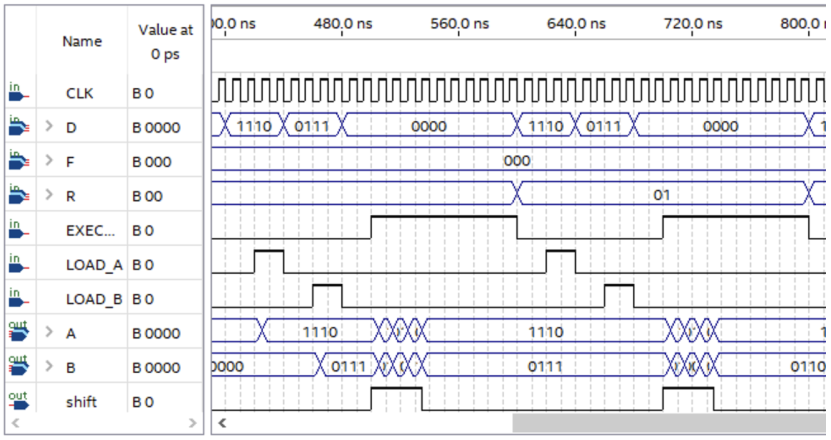

Figure 13: wave of 400-800ns

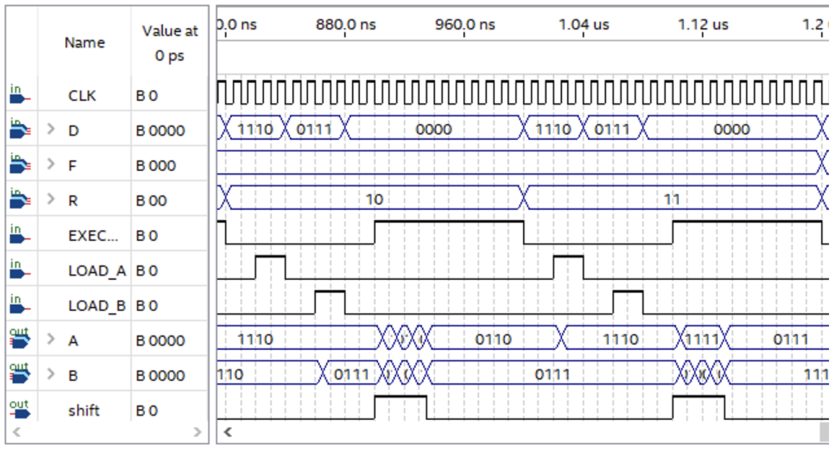

Figure 14: wave of 800-1200ns

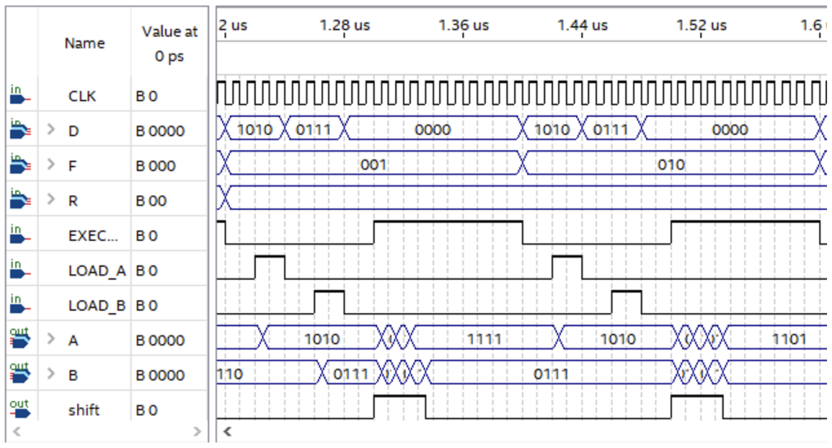

Figure 15: wave of 1200-1600ns

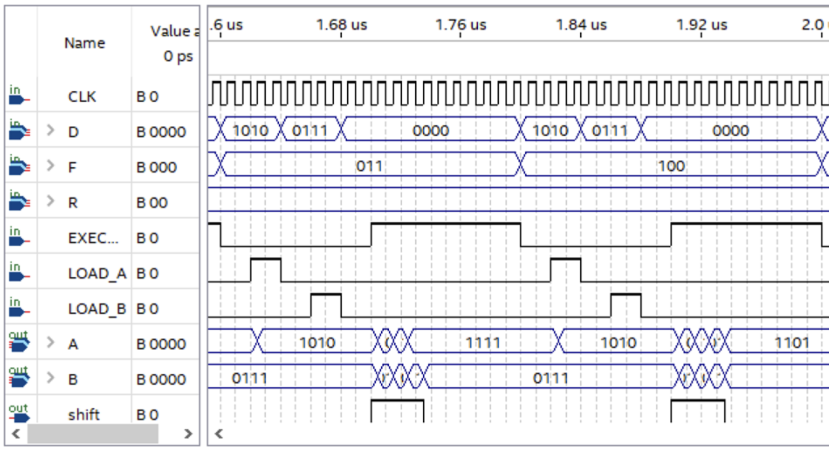

Figure 16: wave of 1600-2000ns

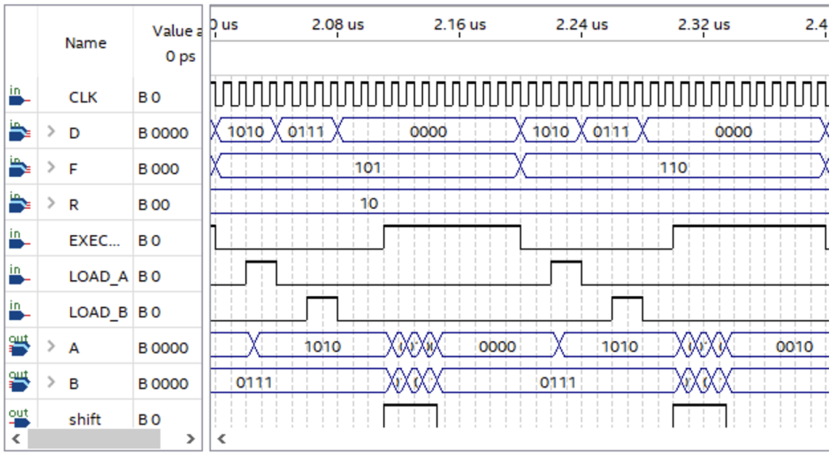

Figure 17: wave of 2000-2400ns

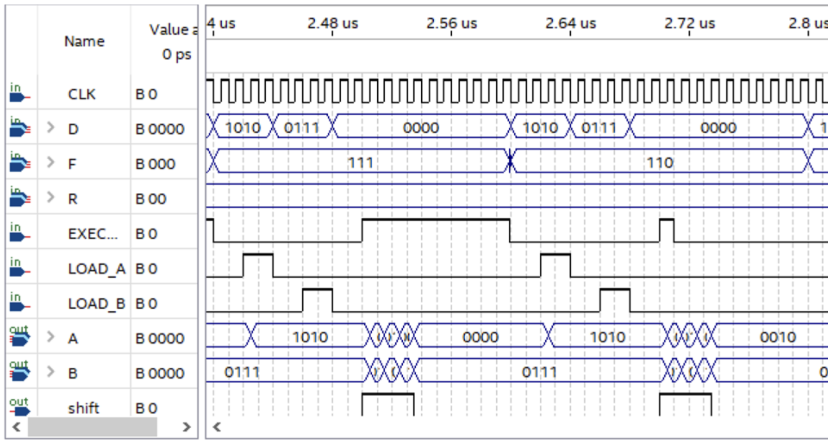

Figure 18: wave of 2400-2800ns